DFSORT PTF UN90139 Year 2000 and Performance Enhancements

April, 1996

Frank L. Yaeger

DFSORT Team
IBM Storage Systems Division
San Jose, California
VNET: YAEGER at SJFEVMX
IBMMAIL: USIB257Z at IBMMAIL
Internet: fyaeger@vnet.ibm.com

DFSORT/MVS Web Site

Come visit DFSORT/MVS on the World Wide Web at URL:

http://www.storage.ibm.com/storage/software/sort/srtmhome.htm

Abstract

DFSORT R13 **PTF UN90139** is available now. This PTF delivers DFSORT **Year 2000 features** you can use to to sort, merge, and transform a wide variety of dates with **two-digit years** according to a specified sliding or fixed century window. It also offers improved sorting performance and increases the maximum amount of data DFSORT can sort.

This package contains a detailed discussion of DFSORT′s new Year 2000 features and examples of the control statements needed to order and transform all kinds of character, zoned decimal and packed decimal date fields. It concludes with an overview of the performance and capacity enhancements.

Contents

DFSORT PTF UN90139 Year 2000 and Performance Enhancements

DFSORT R13 **PTF UN90139** is available now. This PTF:

- Delivers **Year 2000 features** you can use to sort, merge, and transform a wide variety of dates with **two-digit years**, according to a specified sliding or fixed century window.
- Improves sorting performance.
- Increases the maximum amount of data DFSORT can sort.
- Offers floating point data summation capabilities.

This package contains a detailed discussion of DFSORT′s new Year 2000 features and examples of the control statements needed to order and transform all kinds of character, zoned decimal and packed decimal date fields. It concludes with an overview of the performance and capacity enhancements.

For information on the many features of DFSORT mentioned in this package such as OUTFIL and ICETOOL, see one or more of the following:

- *DFSORT Application Programming Guide (SC33-4035)*
- The DFSORT home page at URL:

  http://www.storage.ibm.com/storage/software/sort/srtmhome.htm
- The **SORTBE13** ″Beyond Sorting″ package available from MKTTOOLS or via anonymous FTP to:

  lscftp.kgn.ibm.com/pub/mvs/docs/

Year 2000 Features for Two-Digit Year Dates

The widespread use of two-digits to represent years can result in significant data processing problems. For example, the normal ordering of 00 before 99 is incorrect when 00 represents 2000 and 99 represents 1999. DFSORT′s Year 2000 features provide tools that can help you correctly process a wide variety of dates for this century and the coming ones.

DFSORT R13 PTF UN90139 enhances DFSORT′s Year 2000 capabilities by providing the ability to sort, merge and transform character and packed dates with two-digit years according to a specified sliding or fixed century window. New **Y2C**, **Y2Z**, **Y2P**, and **Y2D** formats, in conjunction with a new **Y2PAST** installation and run-time option, allow you to handle two-digit year data in the following ways:

- Set the appropriate century window for your applications. For example, set a century window of 1915-2014 or 1950-2049.
- Order two-digit character, zoned decimal, packed decimal or decimal year data, according to the century window, using DFSORT′s SORT and MERGE control statements. For example, order 96 (representing 1996) before 00 (representing 2000) in ascending sequence, or order 00 before 96 in descending sequence.
- Transform two-digit character, zoned decimal, packed decimal or decimal year data to four-digit character year data, according to the century window, using DFSORT′s OUTFIL control statement. For example, transform 99 to 1999 and 04 to 2004.

A new **PD0** format allows you to order and transform parts of packed decimal fields (for example, month and day in date fields) using SORT, MERGE and OUTFIL.

These DFSORT enhancements allow you to continue to use two-digit year dates for sorting and merging, and help you to change from using two-digit year dates to using four-digit year dates as appropriate.

Set the Century Window for Two-Digit Years

A new **installation** and **run-time** option allows you to specify the **century window** you want DFSORT to use for interpreting two-digit years. A century window spans a 100 year period, such as 1950-2049, and is used to control how the two-digit years 00 to 99 are interpreted. For example, 56 will be interpreted as 1956 if you set the century window to 1950-2049 or as 2056 if you set the century window to 1990-2089.

DFSORT allows you to set either a **fixed** century window or a **sliding** century window as follows:

- **Y2PAST=f** specifies a **fixed** century window starting at f, where f can be any value between 1000 and 3000. For example, Y2PAST=1985 starts the century window at 1985, which always sets the century window to 1985-2084.

  You might want to use a fixed century window for two-digit year data that you started collecting in some particular year (for example, 1985 in this case) or to examine historic or future two-digit year data (for example, 1200-1299 or 2500-2599).

- **Y2PAST=s** specifies a **sliding** century window starting s years before the current year, where s can be any value between 0 and 100. For example, if the current year is 1996, Y2PAST=80 starts the century window at 1996 - 80 = 1916, which sets the century window to 1916-2015. When 1997 rolls around, Y2PAST=80 starts the century window at 1997 - 80 - 1917, which sets the century window to 1917-2016. You can see that Y2PAST=s causes the century window to automatically slide year by year.

  You might want to use a sliding century window for two-digit year data you collect over a rolling 30 year period.

Thus, the Y2PAST option allows you to control how DFSORT interprets the two-digit years 00 to 99 for various sets of data. You can use the Y2PAST **installation** option to set a default site-wide fixed or sliding century window. The shipped IBM installation default is Y2PAST=80 which sets a sliding window starting 80 years before the current year. You can use the Y2PAST **run-time** option to set specific fixed or sliding century windows for different applications.

As an example of setting the century window, both Y2PAST=1915, and Y2PAST=81 used in 1996, give a century window of 1915 to 2014, which results in the following interpretation of two-digit years by DFSORT.

```
yy        Interpreted as
--        --------------
00        2000
14        2014
15        1915
61        1961
62        1962
99        1999
```

Select New Formats for Two-Digit Year Dates

DFSORT provides several **new formats** that help you deal with a wide variety of dates containing two-digit years.

You can use the new **Y2C, Y2Z, Y2P** and **Y2D** formats to identify your two-digit character, zoned decimal, packed decimal and decimal **year** data, respectively. DFSORT will correctly sort, merge and transform two-digit years identified with these formats as if they were four-digit years, using your specified fixed or sliding century window.

You can use the new **PD0** format to identify your two-digit packed **month** data and two-digit and three-digit packed **day** data. DFSORT will correctly sort, merge and transform months and days identified with these formats.

Here are the details of these new formats:

- **Y2C or Y2Z** - two digit, two-byte character or zoned decimal year data

  Y2C and Y2Z are interchangable and can be used for any two-digit year field that can be represented as:

  x**y**x**y**

  where **y** is hexadecimal 0-9 and represents a year digit, and x is hexadecimal 0-F and is ignored.

  For example, 96 represented as X′F9F6′, X′F9C6′ and X′0906′ can all be handled by Y2C or Y2Z format.

  Thus, you can use Y2C and Y2Z for two-digit years in many types of dates such as C′**yy**′, C′mm/dd/**yy**′, C′**yy**.mm.dd′, Z′ddmm**yy**, Z′**yy**ddd′, C′ddd**yy**′ and C′**yy**-mm′.

- **Y2P** - two-digit, two-byte packed decimal year data

  Y2P can be used for any two-digit year field that can be represented as:

  x**yy**x

  where **y** is hexadecimal 0-9 and represents a year digit, and x is hexadecimal 0-F and is ignored.

  For example, 96 represented as X′096C′, X′096F′ and X′8962′ can all be handled by Y2P format.

  Thus, you can use Y2P for two-digit years in many types of dates such as P′**yy**′, P′mmdd**yy**′, P′**yy**mmdd′, P′ddmm**yy**, P′ddd**yy**′ and P′mm**yy**′.

- **Y2D** - two-digit, one-byte decimal year data

  Y2D can be used for any two-digit year field that can be represented as:

  yy

  where **y** is hexadecimal 0-9 and represents a year digit.

  For example, 96 represented as X′96′ can be handled by Y2D format.

  Thus, you can use Y2D for two-digit years in many types of dates such as X′**yy**′ and P′**yy**ddd′.

- **PD0** - packed decimal with sign and first digit ignored

  PD0 can be used for any two- to eight-byte packed decimal field you want to treat as:

  x**ddd...d**s

  where x is hexadecimal 0-F and is ignored, **d** is hexadecimal 0-9 and represents a decimal digit, and s is hexadecimal 0-F and is ignored.

  For example, 12 represented as X′012C′, X′012D′, X′912C′ and X′3127′ can all be handled by PD0 format, whereas with PD format, these values would be treated as +12, -12, +912 and -312, respectively.

  Thus, PD0 can be used for parts of PD fields such as **mm**, **dd**, **mmdd** and **ddmm** in P′**mmdd**yy′, P′**ddmm**yy′, P′yy**mmdd**′ and P′yy**mm**′.

Sort and Merge on Two-Digit Year Dates

You can use DFSORT′s new formats (Y2C, Y2Z, Y2P, Y2D and PD0) to correctly sort and merge many different kinds of character, zoned decimal and packed decimal dates, including those with two-digit years, according to the century window you specify.

We′ll go through examples here to help you understand how to set up the sort and merge control statements for some typical date fields. But to save you the trouble of reinventing the wheel, a fairly complete list of examples of SORT statements for date fields is provided in "Examples of How to Sort All Kinds of Date Fields" on page 15.

Basics

You always want to order dates by **four-digit year**, then **month** (if present) and then **day** (if present). So if you have a date in the form mmddyy, you would order it as yyyymmdd, and if you have a date in the form yyddd, you would order it as yyyyddd.

For each sort or merge application involving two-digit years, you need to decide whether to use the site′s default century window (CW) or set a different century window for the application using the Y2PAST run-time option.

Although a date field can start anywhere in a record and can be only one of a number of fields you are ordering on, for the sake of simplicity we will assume in the examples below that you are only sorting on one date field which starts in position 1.

Character and Zoned Decimal Dates

Character and zoned decimal dates can be represented as follows in hexadecimal:

```
Character:      FdFd...Fd
Zoned decimal:  FdFd...Cd
```

where **d** is a decimal digit (0-9). For zoned decimal, the last byte contains a C to indicate a positive sign, but for the purpose of sorting or merging, the C sign can be ignored and zoned decimal and character can be treated the same.

There are many types of character and zoned decimal dates. Here are just a few: C′mm/dd/yy′, Z′yymmdd′, C′yyddd′, C′dd.mm.yy′, C′mmyy′.

To sort or merge character or zoned decimal dates, use **Y2C** format for the two-digit year and use **BI** format for the month and the day. A fairly complete list of examples of SORT statements for date fields is provided in "Examples of How to Sort All Kinds of Date Fields" on page 15.

Here is an example of control statements to sort a C′mm/dd/yy′ date field.

```
* Set 1962 to 2061 as the fixed century window
 OPTION Y2PAST=1962
* Sort C¢mm/dd/yy¢ as C¢yyyymmdd¢
 SORT FIELDS=(7,2,Y2C,A,  * sort C¢yy¢ using CW
              1,2,BI,A,   * sort C¢mm¢
              4,2,BI,A)   * sort C¢dd¢
```

Here are some results for the specified century window of 1962-2061.

```
SORTIN input (date field is C¢mm/dd/yy¢)

09/01/96  * represents 09/01/1996 *
11/14/61  * represents 11/14/2061 *
10/03/05  * represents 10/03/2005 *
03/18/62  * represents 03/18/1962 *
10/03/02  * represents 10/03/2002 *

SORTOUT output (date field is C¢mm/dd/yy¢)

03/18/62  * represents 03/18/1962 *
09/01/96  * represents 09/01/1996 *
10/03/02  * represents 10/03/2002 *
10/03/05  * represents 10/03/2005 *
11/14/61  * represents 11/14/2061 *
```

Here is an example of a control statement to sort a Z′yymmdd′ date field.

```
* Use site¢s default century window of 1916-2015
* Sort Z¢yymmdd¢ as Z¢yyyymmdd¢
 SORT FIELDS=(1,2,Y2C,A,  * sort C¢yy¢ using CW
              3,4,BI,A)   * sort C¢mmdd¢
```

Here are some results for the site′s default century window of 1916-2015.

```
SORTIN input (date field is Z¢yymmdd¢)

000428  * represents 2000/04/28 *
620325  * represents 1962/03/25 *
160428  * represents 1916/04/28 *
150622  * represents 2015/06/22 *
990901  * represents 1999/09/01 *
620318  * represents 1962/03/18 *

SORTOUT output (date field is Z¢yymmdd¢)

160428  * represents 1916/04/28 *
620318  * represents 1962/03/18 *
620325  * represents 1962/03/25 *
990901  * represents 1999/09/01 *
000428  * represents 2000/04/28 *
150622  * represents 2015/06/22 *
```

Packed Decimal Dates

Packed decimal data can be represented as follows in hexadecimal:

dd...dC

where **d** is a decimal digit (0-9), and C indicates a positive sign (F is sometimes used for the sign instead of C).

You have to be careful figuring out the positions of the month, day and year in packed decimal dates because it′s not always obvious what they look like when stored. For example, a **P′mmddyy′** field is actually stored as **X′**0**mmddyyC′**. The C to indicate a positive sign is added at the end, and a 0 is added at the beginning to give an even number of hex digits. Thus, the month, day and year each span more than one byte. Bytes 1-2 contain X′0**mm**d′, bytes 2-3 contain X′m**dd**y′ and bytes 3-4 contain X′d**yy**C′. Note that the month, day and year can all contain extra digits before and after the significant digits (for example, m and y in X′m**dd**y′). DFSORT′s new formats give you the ability to ignore the leading digit and trailing sign when necessary.

There are many types of packed decimal dates. Here are just a few: P′yymmdd′ (X′0yymmddC′), P′mmddyy′ (X′0mmddyyC′), P′ddmmyy′ (X′0ddmmyyC′), P′yyddd′ (X′yydddC′), P′dddyy′ (X′dddyyC′), P′yymm′ (X′0yymmC′).

To sort or merge packed decimal dates, use **Y2P** format for a two-byte, two-digit year (for example, X′d**yy**C′ or X′0**yy**m′) or **Y2D** for a one-byte, two-digit year (X′**yy**′) year. For the month and the day, you can generally use **BI** format, but in some cases **PD0** format is needed. A fairly complete list of examples of SORT statements for date fields is provided in “Examples of How to Sort All Kinds of Date Fields” on page 15.

Here is an example of a control statement to sort a P′mmddyy′ (X′0mmddyyC′) date field in **descending** order.

```
* Use site¢s default century window of 1916-2015
* Sort P¢mmddyy¢ (X¢0mmddyyC¢) as P¢yyyymmdd¢
 SORT FIELDS=(3,2,Y2P,D,  * sort X¢dyyC¢ as X¢yy¢ using CW
              1,3,BI,D)   * sort X¢0mmddC¢
```

Here are some results for the site′s default century window of 1916-2015. Remember that the records were sorted in **descending** order.

```
 SORTIN input (date field is P¢mmddyy¢ - shown as X¢0mmddyyC¢)

 0042800C  * represents 04/28/2000 *
 0032562C  * represents 03/25/1962 *
 0042816C  * represents 04/28/1916 *
 0062215C  * represents 06/22/2015 *
 0110199C  * represents 11/01/1999 *
 0031862C  * represents 03/18/1962 *

 SORTOUT output (date field is P¢mmddyy¢ - shown as X¢0mmddyyC¢)

 0062215C  * represents 06/22/2015 *
 0042800C  * represents 04/28/2000 *
 0110199C  * represents 11/01/1999 *
 0032562C  * represents 03/25/1962 *
 0031862C  * represents 03/18/1962 *
 0042816C  * represents 04/28/1916 *
```

Here is an example of control statements to sort a C′yyddd′ (X′yydddC′) date field.

```
* Set 1962 to 2061 as the fixed century window
 OPTION Y2PAST=1962
* Sort P¢yyddd¢ (X¢yydddC¢) as P¢yyyyddd¢
 SORT FIELDS=(1,1,Y2D,A,  * sort X¢yy¢ using CW
              2,2,BI,A)   * sort X¢dddC¢
```

Here are some results for the specified century window of 1962-2061.

```
SORTIN input (date field is P¢yyddd¢ - shown as X¢yydddC¢)

002001C  * represents 2002/001 *
061300C  * represents 2061/300 *
096221C  * represents 1996/221 *
005128C  * represents 2005/128 *
062025C  * represents 1962/025 *

SORTOUT output (date field is P¢yyddd¢ - shown as X¢yydddC¢)

062025C  * represents 1962/025 *
096221C  * represents 1996/221 *
002001C  * represents 2002/001 *
005128C  * represents 2005/128 *
061300C  * represents 2061/300 *
```

Here is an example of control statements to merge a P′ddmmyy′ (X′0ddmmyyC′) date field.

```
* Set sliding century window starting 20 years in the past
 OPTION Y2PAST=20
* Merge P¢ddmmyy¢ (X¢0ddmmyyC¢) as P¢yyyymmdd¢
 MERGE FIELDS=(3,2,Y2P,A, * sort X¢myyC¢ as X¢yy¢ using CW
               2,2,PD0,A, * sort X¢dmmy¢ as X¢mm¢
               1,2,BI,A)  * sort X¢0ddm¢
```

Y2PAST=20 sets a sliding century window starting 20 years in the past, so in 1996 the century window would be 1976-2075. Here are some results for that century window.

```
SORTIN01 input (date field is P¢ddmmyy¢ - shown as X¢0ddmmyyC¢)

0291076C  * represents 29/10/1976 *
0031299C  * represents 03/12/1999 *
0210473C  * represents 21/04/2073 *
0011275C  * represents 01/12/2075 *

SORTIN02 input (date field is P¢ddmmyy¢ - shown as X¢0ddmmyyC¢)

0281076C  * represents 28/10/1976 *
0170596C  * represents 17/05/1996 *
0170573C  * represents 17/05/2073 *

SORTOUT output (date field is P¢ddmmyy¢ - shown as X¢0ddmmyyC¢)

0281076C  * represents 28/10/1976 *
0291076C  * represents 29/10/1976 *
0170596C  * represents 17/05/1996 *
0031299C  * represents 03/12/1999 *
0210473C  * represents 21/04/2073 *
0170573C  * represents 17/05/2073 *
0011275C  * represents 01/12/2075 *
```

Transform Two-Digit Year Dates to Four-Digit Year Dates

You can use DFSORT′s new formats (Y2C, Y2Z, Y2P, Y2D and PD0) to transform many different kinds of dates, including those with two-digit years, according to the century window you specify. Using the **OUTREC operand** of the **OUTFIL** statement, you can:

- convert two-digit year fields in character, zoned decimal and packed decimal format to four-digit year fields in character format (for example, convert C′yy′ to C′yyyy′)

- convert month and day fields in zoned decimal and packed decimal format to month and day fields in character format (for example, convert P′ddmmyy′ to C′ddmmyyyy′)
- rearrange the order of the year, day and month (for example, convert Z′mmddyy′ to C′yyyymmdd′)
- insert separators between the year, month and day (for example, convert P′yyddmm′ to C′dd.mm.yyyy′ or C′mm/dd/yyyy′)

We′ll go through examples here to help you understand how to set up the OUTFIL control statements for transforming some typical date fields. But to save you the trouble of reinventing the wheel, a fairly complete list of examples of OUTFIL statements for transforming date fields is provided in "Examples of How to Transform All Kinds of Date Fields" on page 21.

You can use the OUTFIL date transformation techniques shown here for a variety of tasks such as to create new data sets with four-digit years or to produce reports containing tranformed dates.

Basics

For each application involving transforming two-digit years, you need to decide whether to use the site′s default century window (CW) or set a different century window for the application using the Y2PAST run-time option.

For zoned decimal and packed decimal date fields, you can use OUTFIL′s editing capabilities to change the month to two characters and the day to two or three characters. Just specify the format of the month and day (ZD, PD, or PD0) and OUTFIL′s **M11 edit mask**, which produces two readable character digits without a sign (for example, ZD,M11 for Z′25′ = X′F2C5′ will produce C′25′). M11 is just one of OUTFIL′s 26 predefined edit masks that allow you to convert BI, FI, PD, ZD and FS values to character values of various formats.

Although a date field can start anywhere in a record and can be only one of a number of fields you are working with, for the sake of simplicity we will assume in the examples below (unless stated otherwise) that you are only transforming one date field which starts in position 1.

Character Dates

Character dates can be represented as follows in hexadecimal:

FdFd...Fd

where **d** is a decimal digit (0-9).

There are many types of character dates. Here are just a few: C′mm/dd/yy′, C′yymmdd′, C′yyddd′, C′dd.mm.yy′, C′cyymmdd′.

To transform character dates, use **Y2C** format for the two-digit year and **copy** the month and the day.

Here is an example of control statements to transform a C′mm/dd/yy′ date field into a C′yyyy-mm-dd′ date field.

```
* Set 1962 to 2061 as the fixed century window
 OPTION COPY,Y2PAST=1962
* Transform C¢mm/dd/yy¢ to C¢yyyy-mm-dd¢
 OUTFIL OUTREC=(7,2,Y2C, * change C¢yy¢ to C¢yyyy¢ using CW
                C¢-¢,    * insert dash
                1,2,     * copy C¢mm¢
                C¢-¢,    * insert dash
                4,2)     * copy C¢dd¢
```

Here are some results for the specified century window of 1962-2061.

```
SORTIN input (date field is C¢mm/dd/yy¢)

09/01/96
11/14/61
10/03/05
03/18/62
10/03/02

SORTOUT output (date field is C¢yyyy-mm-dd¢)

1996-09-01
2061-11-14
2005-10-03
1962-03-18
2002-10-03
```

If you have a **century indicator** in a character date, you can use it to convert the two-digit year to a four-digit year without using the century window (of course, you could ignore the century indicator and use the century window instead if you want to). Use the **lookup and change** feature of the OUTFIL statement to convert the century indicator to C′19′, C′20′, and so on.

Here is an example of control statements to transform a C′cyymmdd′ date field into a C′yyyymmdd′ date field. For this example, we′ll assume the century indicator is C′0′ for a 19yy date, C′1′ for a 20yy date or C′2′ for a 21yy date.

```
* Sort C¢cyymmdd¢
 SORT FIELDS=(1,7,BI,A)          * sort C¢cyymmdd¢
* Transform C¢cyymmdd¢ to C¢yyyymmdd¢
 OUTFIL OUTREC=(1,1,CHANGE=(2,   * change C¢c¢ as follows:
                   C¢0¢,C¢19¢,  *   C¢0¢ to C¢19¢
                   C¢1¢,C¢20¢,  *   C¢1¢ to C¢20¢
                   C¢2¢,C¢21¢), *   C¢2¢ to C¢21¢
              2,6)              * copy C¢yymmdd¢
```

Here are some results.

```
SORTIN input (date field is C¢cyymmdd¢)

1122505
0960215
2102818
0991122

SORTOUT output (date field is C¢yyyymmdd¢)

19960215
19991122
20122505
21102818
```

Zoned Decimal Dates

Zoned decimal dates can be represented as follows in hexadecimal:

F**d**F**d**...C**d**

where **d** is a decimal digit (0-9). The last byte contains a C to indicate a positive sign. So, if the last digit is 5, it will actually be X′C5′ which is C′E′. **Y2C** will ignore the C, so if the zoned decimal date ends with a two-digit year, you can treat it as if it were a character date. But if the zoned decimal date ends with the month or day, copying it as you would for a character date will give you a C′E′ instead of a C′5′. Use **ZD** format and OUTFIL′s **M11 edit mask** for the month or day (whichever is last) to make sure you get C′5′ instead of C′E′.

Here are some of the zoned decimal dates that can be transformed like character dates: Z′mmddyy′, Z′ddmmyy′, Z′dddyy′, Z′mmyy′. And here are some of the zoned decimal dates that must be transformed as zoned decimal dates: Z′yymmdd′, Z′yyddmm′, Z′yyddd′, Z′yymm′. A fairly complete list of examples of OUTFIL statements for transforming date fields is provided in "Examples of How to Transform All Kinds of Date Fields" on page 21.

Here is an example of control statements to create a **new** data set with a character date field (C′mm/dd/yyyy′) containing four-digit years from an **old** data set with a zoned decimal date field (Z′yymmdd′) containing two-digit years. Assume that the old data set has a Z′yymmdd′ date field in position 21 and is fixed format with a logical record length of 200. Since the 6 digit zoned decimal date field will become a 10 character date field, the new data set will have a logical record length of 204.

```
* Use site¢s default century window of 1916-2015
 OPTION COPY
* Transform Z¢yymmdd¢ to C¢mm/dd/yyyy¢ in the new data set
 OUTFIL FNAMES=NEW,
    OUTREC=(1,20,         * copy positions 1-20
            23,2,         * copy Z¢mm¢ = C¢mm¢
            C¢/¢,         * insert slash
            25,2,ZD,M11,  * edit X¢FdCd¢ to C¢dd¢
            C¢/¢,         * insert slash
            21,2,Y2C,     * change Z¢yy¢ to C¢yyyy¢ using CW
            27,174)       * copy the rest of the record
```

Note that the C sign in the day field makes it necessary to treat it as a ZD field and edit it with OUTFIL′s M11 mask.

Here are some results for the site′s default century window of 1916-2015.

```
 SORTIN input date field in position 21
 (Z¢yymmdd¢ - shown as C¢yymmdd¢ - note last character)

 96040E
 02121C
 88072B

 NEW output date field in position 21 (C¢mm/dd/yyyy¢)

 04/05/1996
 12/13/2002
 07/22/1988
```

Packed Decimal Dates

Packed decimal data can be represented as follows in hexadecimal:

dd...dC

where **d** is a decimal digit (0-9), and C indicates a positive sign (F is sometimes used for the sign instead of C).

You have to be careful figuring out the positions of the month, day and year in packed decimal dates because it′s not always obvious what they look like when stored. For example, a **P′mmddyy′** field is actually stored as **X′**0**mmddyyC′**. The C to indicate a positive sign is added at the end, and a 0 is added at the beginning to give an even number of hex digits. Thus, the month, day and year each span more than one byte. Bytes 1-2 contain X′0**mm**d′, bytes 2-3 contain X′m**dd**y′ and bytes 3-4 contain X′d**yy**C′. Note that the month, day and year can all contain extra digits before and after the significant digits (for example, m and y in X′m**dd**y′). DFSORT′s edit masks and new formats give you the the ability to ignore the leading digit and trailing sign when necessary.

There are many types of packed decimal dates. Here are just a few: P′yymmdd′ (X′0yymmddC′), P′mmddyy′ (X′0mmddyyC′), P′ddmmyy′ (X′0ddmmyyC′), P′yyddd′ (X′yydddC′), P′dddyy′ (X′dddyyC′), P′yymm′ (X′0yymmC′).

To transform a packed decimal date, use **Y2P** format for a two-byte, two-digit year (for example, X′d**yy**C′ or X′0**yy**m′) or **Y2D** format for a one-byte, two-digit year (X′**yy**′) year. For the month and the day, use **PD** format if all of the digits in the value are significant (for example, X′**ddd**C′ or X′**ddd**y′). Otherwise, use **PD0** format to ignore the first digit and/or sign (for example, X′m**dd**y′ or X′0**dd**m′). Use the **M11 edit mask** to produce two characters for the month and two or three characters for the day. A fairly complete list of examples of OUTFIL statements for transforming date fields is provided in “Examples of How to Transform All Kinds of Date Fields” on page 21.

Here is an example of control statements to transform a P′ddmmyy′ (X′0ddmmyyC′) date field into a C′dd.mm.yyyy′ date field.

```
* Use site¢s default century window of 1916-2015
  SORT FIELDS=(6,4,CH,A)        * sort on key field
* Transform P¢ddmmyy¢ (X¢0ddmmyyC¢) to C¢dd.mm.yyyy¢
  OUTFIL OUTREC=(1,2,PD0,M11,   * edit X¢0ddm¢ to C¢dd¢
                 C¢.¢,          * insert period
                 2,2,PD0,M11,   * edit X¢dmmy¢ to C¢mm¢
                 C¢.¢,          * insert period
                 3,2,Y2P,       * change X¢myyC¢ to C¢yyyy¢ using CW
                 5,20)          * copy the rest of the record
```

Here are some results for the site′s default century window of 1916-2015.

```
 SORTIN input (date field is P¢ddmmyy¢ - shown as X¢0ddmmyyC¢)

 0011199C KEY5
 0280400C KEY2
 0250362C KEY1
 0280416C KEY3
 0220615C KEY6
 0180362C KEY4

 SORTOUT output (date field is C¢dd.mm.yyyy¢)

 25.03.1962 KEY1
 28.04.2000 KEY2
 28.04.1916 KEY3
 18.03.1962 KEY4
 01.11.1999 KEY5
 22.06.2015 KEY6
```

Here is an example of control statements to transform a C′yyddd′ (X′yydddC′) date field into a C′yyyyddd′ date field.

```
* Set 1962 to 2061 as the fixed century window
 OPTION Y2PAST=1962,COPY
* Transform P¢yyddd¢ (X¢yydddC¢) to C¢yyyyddd¢
 OUTFIL OUTREC=(1,1,Y2D,      * change X¢yy¢ to C¢yyyy¢ using CW
                2,2,PD,M11)   * edit X¢dddC¢ to C¢ddd¢
```

Here are some results for the specified century window of 1962-2061.

```
SORTIN input (date field is P¢yyddd¢ - shown as X¢yydddC¢)

002001C
061300C
096221C
005128C
062025C

SORTOUT output (date field is C¢yyyyddd¢)

1962025
1996221
2002001
2005128
2061300
```

If you have a **century indicator** in a packed decimal date, you can use it to convert the two-digit year to a four-digit year without using the century window (of course, you could ignore the century indicator and use the century window instead if you want to). Use the **lookup and change** feature of the OUTFIL statement to convert the century indicator to C′19′, C′20′, and so on.

Here is an example of control statements to transform a P′cyymmdd′ (X′cyymmddC′) date field into a C′mm/dd/yyyy′ date field. For this example, we′ll assume the century indicator is 0 for a 19yy date, 1 for a 20yy date or 2 for a 21yy date.

```
 OPTION COPY
* Transform P¢cyymmdd¢ (X¢cyymmddC¢) to C¢mm/dd/yyyy¢
 OUTFIL OUTREC=(2,2,PD0,M11,            * edit X¢ymmd¢ to C¢mm¢
               C¢/¢,                    * insert slash
               3,2,PD0,M11,             * edit X¢mddC¢ to C¢dd¢
               C¢/¢,                    * insert slash
               1,1,CHANGE=(2,           * change c as follows:
                   B¢0000....¢,C¢19¢,   *   0 to C¢19¢
                   B¢0001....¢,C¢20¢,   *   1 to C¢20¢
                   B¢0010....¢,C¢21¢),  *   2 to C¢21¢
               1,2,PD0,M11)             * edit X¢cyym¢ to C¢yy¢
```

Here are some results.

```
SORTIN input (date field is P¢cyymmdd¢ - shown as X¢cyymmddC¢)

2120818C
0960215C
1000505C
0991122C

SORTOUT output (date field is C¢mm/dd/yyyy¢)

08/18/2112
02/15/1996
05/05/2000
11/22/1999
```

Report on Two-Digit Year Dates

You can use OUTFIL to transform dates so that they can be used in reports.

For a given application, you can either:

- use **OUTFIL** to transform the dates and create the report all at once, or
- use **OUTFIL** to transform the dates and then use **ICETOOL** to create the report.

Choosing to create the report with ICETOOL vs ICETOOL is a matter of the **time** you want to spend on the report and how much **control** you need over the way it looks.

OUTFIL Reports

Transformed dates can easily be incorporated directly into **OUTFIL reports** by including them in the **OUTREC operand**.

Here is an example of JCL and control statements to produce an OUTFIL report with a P′yymmdd′ date field transformed into a C′mm/dd/yyyy′ date field.

```
//REPORT1 EXEC  PGM=ICEMAN
//SYSOUT    DD  SYSOUT=*
//IN DD DSN=TRANS.INPUT,DISP=SHR
//RPT1 DD SYSOUT=*
//SYSIN DD *
* Use century window of 1990-2089
 OPTION Y2PAST=1990
* Sort on P¢yymmdd¢
 SORT FIELDS=(21,2,Y2P,A,2,3,BI,A)
* Create report with P¢yymmdd¢ (X¢0yymmddC¢) transformed
* to C¢mm/dd/yyyy¢
 OUTFIL FNAMES=RPT1,
   HEADER2=(2:C¢Transaction Report for ¢,DATE=(MD4/),2/,
           2:C¢Date      ¢,17:C¢Item Code¢,31:C¢Price ($)¢,/,
           2:C¢----------¢,17:C¢---------¢,31:C¢---------¢),
   OUTREC=(2:22,2,PD0,M11,C¢/¢,     * mm/
            23,2,PD0,M11,C¢/¢,     * dd/
            21,2,Y2P,              * yyyy
          17:31,5,                 * Item code
          31:5,4,PD,M18,LENGTH=9,  * Price
          120:X)                   * Pad with blanks
/*
```

Notice that **DATE=(MD4/)** tells OUTFIL to produce a **mm/dd/yyyy** format date in the title. In fact, OUTFIL lets you specify that the month (M), day (D), two-digit (Y) or four-digit year (4) and any separator character you select, be in any order you like in the report title.

Here is what the report might look like.

```
 Transaction Report for 04/09/1996

 Date           Item Code     Price ($)
 ----------     ---------     ---------
 11/21/1999     ZZ723          1,203.25
 12/13/1999     AX123            623.87
 01/08/2000     BR321            318.05
 01/16/2000     QZ520          1,621.36
```

ICETOOL Reports

Transformed dates can easily be used in **ICETOOL reports** by using a **COPY or SORT operator** with **OUTFIL OUTREC** followed by a **DISPLAY operator**.

Here is an example of JCL and control statements to produce an ICETOOL report with a P′yymmdd′ date field transformed into a C′mm/dd/yyyy′ date field.

```
//REPORT2 EXEC  PGM=ICETOOL
//TOOLMSG   DD  SYSOUT=*
//DFSMSG    DD  SYSOUT=*
//IN DD DSN=TRANS.INPUT,DISP=SHR
//TEMP DD DSN=&T1,DISP=(,PASS),SPACE=(TRK,(5,5)),UNIT=SYSDA
//RPT2 DD SYSOUT=*
//TOOLIN DD *
* Create temporary file with transformed date and
* other needed fields
 SORT FROM(IN) USING(TRAN)
* Create report
 DISPLAY FROM(TEMP) LIST(RPT2) BLANK -
   TITLE(¢Transaction Report for¢) DATE(MD4/) -
   HEADER(¢Date¢) ON(1,10,CH) -
   HEADER(¢Item Code¢) ON(15,5,CH) -
   HEADER(¢Price ($)¢) ON(25,4,PD,C1)
/TRANCNTL DD *
* Use century window of 1990-2089
  OPTION Y2PAST=1990
* Sort on P¢yymmdd¢
  SORT FIELDS=(21,2,Y2P,A,2,3,BI,A)
* Transform P¢yymmdd¢ (X¢0yymmddC¢) to C¢mm/dd/yyyy¢
* and copy needed fields
  OUTFIL FNAMES=TEMP,
    OUTREC=(1:22,2,PD0,M11,C¢/¢,  * mm/
              23,2,PD0,M11,C¢/¢,  * dd/
              21,2,Y2P,           * yyyy
           15:31,5,               * Item code
           25:5,4)                * Price
/*
```

Notice that **DATE(MD4/)** tells ICETOOL to produce a **mm/dd/yyyy** format date in the title. In fact, ICETOOL lets you specify that the month (M), day (D), two-digit (Y) or four-digit year (4) and any separator character you select, be in any order you like in the report title.

Here is what the report might look like.

```
Transaction Report for        04/09/1996

Date         Item Code    Price ($)
----------   ---------   ----------
11/21/1999   ZZ723         1,203.25
12/13/1999   AX123           623.87
01/08/2000   BR321           318.05
01/16/2000   QZ520         1,621.36
```

Examples of How to Sort All Kinds of Date Fields

Listed here are **SORT statements** you can use with DFSORT for all kinds of date fields. "Sort and Merge on Two-Digit Year Dates" on page 4 provides details on the date ordering techniques used in these examples.

For the sake of simplicity, each example sorts on a single character, zoned decimal or packed decimal date field. In general, these fields can be represented as follows in hexadecimal:

- Character - F**d**F**d**...F**d**
- Zoned decimal - F**d**F**d**...C**d**
- Packed decimal - **dd**...**d**C

where **d** is a decimal digit (0-9) and C indicates a positive sign (F is sometimes used for the sign instead of C).

The elements of the various dates are as follows:

- **mm** - month
- **dd** - day of month
- **ddd** - day of year
- **yy** - two-digit year
- **yyyy** - four-digit year
- **c** - century indicator - 0 for 19, 1 for 20, and so on
- **s** - separator, for example, "/" (slash), "." (period) or "-" (dash)
- **ww** - week
- **d** - day of week

The examples assume that the date field starts in position 1, but the relative position of each element is shown under each example using **p** and **p+n** notation, to make it easier to extrapolate the example to the general case.

Index

Use the following ″index″ to find any example you are interested in:

(01) C¢yymmdd¢	(01) Z¢yymmdd¢	(02) P¢yymmdd¢	
(03) C¢mmddyy¢	(03) Z¢mmddyy¢	(04) P¢mmddyy¢	
(05) C¢ddmmyy¢	(05) Z¢ddmmyy¢	(06) P¢ddmmyy¢	
(07) C¢yyddmm¢	(07) Z¢yyddmm¢	(08) P¢yyddmm¢	
(09) C¢yyddd¢	(09) Z¢yyddd¢	(10) P¢yyddd¢	
(11) C¢dddyy¢	(11) Z¢dddyy¢	(12) P¢dddyy¢	
(13) C¢yysmmsdd¢	(14) C¢mmsddsyy¢	(15) C¢ddsmmsyy¢	(16) C¢yysddsmm¢
(17) C¢yyyymmdd¢	(17) Z¢yyyymmdd¢	(18) P¢yyyymmdd¢	
(19) C¢ddmmyyyy¢	(19) Z¢ddmmyyyy¢	(20) P¢ddmmyyyy¢	
(21) C¢cyymmdd¢	(21) Z¢cyymmdd ¢	(22) P¢cyymmdd¢	
(23) C¢mmddyyc¢	(23) Z¢mmddyyc¢	(24) X¢0mmddyyc¢	
(25) C¢yymm¢	(25) Z¢yymm¢	(26) P¢yymm¢	
(27) C¢mmyy¢	(27) Z¢mmyy¢	(28) P¢mmyy¢	
(29) C¢yysmm¢	(30) C¢mmsyy¢		
(31) C¢yywwd¢	(31) Z¢yywwd¢	(32) P¢yywwd¢	

Examples

1. C'yymmdd' or Z'yymmdd'

   ```
   SORT FIELDS=(1,2,Y2C,A,3,4,BI,A)

     yy at p, mm at p+2, dd at p+4
   ```

2. P'yymmdd' (X'0yymmddC')

   ```
   SORT FIELDS=(1,2,Y2P,A,2,3,BI,A)

     yy at p, mm at p+1, dd at p+2
   ```

3. C'mmddyy' or Z'mmddyy'

   ```
   SORT FIELDS=(5,2,Y2C,A,1,4,BI,A)

     yy at p+4, mm at p, dd at p+2
   ```

4. P'mmddyy' (X'0mmddyyC')

   ```
   SORT FIELDS=(3,2,Y2P,A,1,3,BI,A)

     yy at p+2, mm at p, dd at p+1
   ```

5. C'ddmmyy' or Z'ddmmyy'

   ```
   SORT FIELDS=(5,2,Y2C,A,3,2,BI,A,1,2,BI,A)

     yy at p+4, mm at p+2, dd at p
   ```

6. P'ddmmyy' (X'0ddmmyyC')

   ```
   SORT FIELDS=(3,2,Y2P,A,2,2,PD0,A,1,2,BI,A)

     yy at p+2, mm at p+1, dd at p
   ```

7. C'yyddmm' or Z'yyddmm'

   ```
   SORT FIELDS=(1,2,Y2C,A,5,2,BI,A,3,2,BI,A)

     yy at p, mm at p+4, dd at p+2
   ```

8. P'yyddmm' (X'0yyddmmC')

   ```
   SORT FIELDS=(1,2,Y2P,A,3,2,PD0,A,2,2,BI,A)

     yy at p, mm at p+2, dd at p+1
   ```

9. C'yyddd' or Z'yyddd'

   ```
   SORT FIELDS=(1,2,Y2C,A,3,3,BI,A)

     yy at p, ddd at p+2
   ```

10. P'yyddd' (X'yydddC')

    ```
    SORT FIELDS=(1,1,Y2D,A,2,2,BI,A)

      yy at p, ddd at p+1
    ```

11. C′dddyy′ or Z′dddyy′

```
SORT FIELDS=(4,2,Y2C,A,1,3,BI,A)
```

```
yy at p+3, ddd at p
```

12. P′dddyy′ (X′dddyyC′)

```
SORT FIELDS=(2,2,Y2P,A,1,2,BI,A)
```

```
yy at p+1, ddd at p
```

13. C′yysmmsdd′

```
SORT FIELDS=(1,2,Y2C,A,4,2,BI,A,7,2,BI,A)
```

```
yy at p, mm at p+3, dd at p+6
```

14. C′mmsddsyy′

```
SORT FIELDS=(7,2,Y2C,A,1,2,BI,A,4,2,BI,A)
```

```
yy at p+6, mm at p, dd at p+3
```

15. C′ddsmmsyy′

```
SORT FIELDS=(7,2,Y2C,A,4,2,BI,A,1,2,BI,A)
```

```
yy at p+6, mm at p+3, dd at p
```

16. C′yysddsmm′

```
SORT FIELDS=(1,2,Y2C,A,7,2,BI,A,4,2,BI,A)
```

```
yy at p, mm at p+6, dd at p+3
```

17. C′yyyymmdd′ or Z′yyyymmdd′

```
SORT FIELDS=(1,8,BI,A)
```

```
yyyy at p, mm at p+4, dd at p+6
```

18. P′yyyymmdd′ (X′0yyyymmddC′)

```
SORT FIELDS=(1,5,BI,A)
```

```
yyyy at p, mm at p+2, dd at p+3
```

19. C′ddmmyyyy′ or Z′ddmmyyyy′

```
SORT FIELDS=(5,4,BI,A,3,2,BI,A,1,2,BI,A)
```

```
yyyy at p+4, mm at p+2, dd at p
```

20. P′ddmmyyyy′ (X′0ddmmyyyyC′)

```
SORT FIELDS=(3,3,PD0,A,2,2,PD0,A,1,2,PD0,A)
```

```
yyyy at p+2, mm at p+1, dd at p
```

21. C′cyymmdd′ or Z′cyymmdd′

```
SORT FIELDS=(1,7,BI,A)
```

```
  c at p, yy at p+1, mm at p+3, dd at p+5
```

22. P′cyymmdd′ (X′cyymmddC′)

```
SORT FIELDS=(1,4,BI,A)
```

```
  c at p, yy at p, mm at p+1, dd at p+2
```

23. C′mmddyyc′ or Z′mmddyyc′

```
SORT FIELDS=(7,1,BI,A,5,2,BI,A,1,4,BI,A)
```

```
  c at p+6, yy at p+4, mm at p, dd at p+2
```

24. X′0mmddyyc′

```
SORT FIELDS=(4.4,0.4,BI,A,3,2,PD0,A,1,3,PD0,A)
```

```
  c at p+3.4, yy at p+2, mm at p, dd at p+1
```

25. C′yymm′ or Z′yymm′

```
SORT FIELDS=(1,2,Y2C,A,3,2,BI,A)
```

```
  yy at p, mm at p+2
```

26. P′yymm′ (X′0yymmC′)

```
SORT FIELDS=(1,2,Y2P,A,2,2,BI,A)
```

```
  yy at p, mm at p+1
```

27. C′mmyy′ or Z′mmyy′

```
SORT FIELDS=(3,2,Y2C,A,1,2,BI,A)
```

```
  yy at p+2, mm at p
```

28. P′mmyy′ (X′0mmyyC′)

```
SORT FIELDS=(2,2,Y2P,A,1,2,BI,A)
```

```
  yy at p+1, mm at p
```

29. C′yysmm′

```
SORT FIELDS=(1,2,Y2C,A,4,2,BI,A)
```

```
  yy at p, mm at p+3
```

30. C′mmsyy′

```
SORT FIELDS=(4,2,Y2C,A,1,2,BI,A)
```

```
  yy at p+2, mm at p
```

31. C′yywwd′ or Z′yywwdd′

```
SORT FIELDS=(1,2,Y2C,A,3,3,BI,A)
```

```
yy at p, ww at p+2, d at p+4
```

32. P′yywwd′ (X′yywwdC′)

```
SORT FIELDS=(1,1,Y2D,A,2,2,BI,A)
```

```
yy at p, ww at p+1, d at p+2
```

Examples of How to Transform All Kinds of Date Fields

Listed here are **OUTFIL statements** you can use with DFSORT for all kinds of date fields. "Transform Two-Digit Year Dates to Four-Digit Year Dates" on page 7 provides details on the date transform techniques used in these examples.

For the sake of simplicity, each example transforms a single character, zoned decimal or packed decimal date field. In general, these fields can be represented as follows in hexadecimal:

- Character - F**d**F**d**...F**d**
- Zoned decimal - F**d**F**d**...C**d**
- Packed decimal - **dd**...**d**C

where **d** is a decimal digit (0-9) and C indicates a positive sign (F is sometimes used for the sign instead of C).

The elements of the various dates are as follows:

- **mm** - month
- **dd** - day of month
- **ddd** - day of year
- **yy** - two-digit year
- **yyyy** - four-digit year
- **c** - century indicator - 0 for 19, 1 for 20, and so on
- **s** - separator, for example, ″/″ (slash), ″.″ (period) or ″-″ (dash)
- **ww** - week
- **d** - day of week

The examples assume the following:

- The input date field starts in position 1. The relative position of each element is shown under each example using **p** and **p+n** notation, to make it easier to use the examples as a guide for the general case.
- The input date field is to be transformed to an **output date** field of the form **C′yyyymmdd′**, **C′yyyyddd′**, **C′yyyymm′** or **C′yyyywwd′** as shown in the header for each example. You can use the examples as a guide to producing any output date form you like.
- The output date field is to start in position 1. You can use the column operand (c:) to place the output date field in any position you like.

Index

Use the following ″index″ to find any example you are interested in:

(01) C¢yymmdd¢	(02) Z¢yymmdd¢	(03) P¢yymmdd¢	
(04) C¢mmddyy¢	(04) Z¢mmddyy¢	(05) P¢mmddyy¢	
(06) C¢ddmmyy¢	(06) Z¢ddmmyy¢	(07) P¢ddmmyy¢	
(08) C¢yyddmm¢	(09) Z¢yyddmm¢	(10) P¢yyddmm¢	
(11) C¢yyddd¢	(12) Z¢yyddd¢	(13) P¢yyddd¢	
(14) C¢dddyy¢	(14) Z¢dddyy¢	(15) P¢dddyy¢	
(16) C¢yysmmsdd¢	(17) C¢mmsddsyy¢	(18) C¢ddsmmsyy¢	(19) C¢yysddsmm¢
(20) C¢yyyymmdd¢	(21) Z¢yyyymmdd¢	(22) P¢yyyymmdd¢	
(23) C¢ddmmyyyy¢	(24) Z¢ddmmyyyy¢	(25) P¢ddmmyyyy¢	
(26) C¢cyymmdd¢	(27) Z¢cyymmdd ¢	(28) P¢cyymmdd¢	
(29) C¢mmddyyc¢	(30) Z¢mmddyyc¢	(31) X¢0mmddyyc¢	
(32) C¢yymm¢	(33) Z¢yymm¢	(34) P¢yymm¢	
(35) C¢mmyy¢	(35) Z¢mmyy¢	(36) P¢mmyy¢	
(37) C¢yysmm¢	(38) C¢mmsyy¢		
(39) C¢yywwd¢	(40) Z¢yywwd¢	(41) P¢yywwd¢	

Examples

1. C′yymmdd′ to C′yyyymmdd′

```
OUTFIL OUTREC=(1,2,Y2C,3,4)
```

```
yy at p, mm at p+2, dd at p+4
```

2. Z′yymmdd′ to C′yyyymmdd′

```
OUTFIL OUTREC=(1,2,Y2C,3,2,5,2,ZD,M11)
```

```
yy at p, mm at p+2, dd at p+4
```

3. P′yymmdd′ (X′0yymmddC′) to C′yyyymmdd′

```
OUTFIL OUTREC=(1,2,Y2P,2,3,PD0,M11)
```

```
yy at p, mm at p+1, dd at p+2
```

4. C′mmddyy′ or Z′mmddyy′ to C′yyyymmdd′

```
OUTFIL OUTREC=(5,2,Y2C,1,4)
```

```
yy at p+4, mm at p, dd at p+2
```

5. P′mmddyy′ (X′0mmddyyC′) to C′yyyymmdd′

```
OUTFIL OUTREC=(3,2,Y2P,1,3,PD0,M11)
```

```
yy at p+2, mm at p, dd at p+1
```

6. C′ddmmyy′ or Z′ddmmyy′ to C′yyyymmdd′

```
OUTFIL OUTREC=(5,2,Y2C,3,2,1,2)
```

```
yy at p+4, mm at p+2, dd at p
```

7. P′ddmmyy′ (X′0ddmmyyC′) to C′yyyymmdd′

```
OUTFIL OUTREC=(3,2,Y2P,2,2,PD0,M11,1,2,PD0,M11)
```

```
yy at p+2, mm at p+1, dd at p
```

8. C′yyddmm′ to C′yyyymmdd′

```
OUTFIL OUTREC=(1,2,Y2C,5,2,3,2)
```

```
yy at p, mm at p+4, dd at p+2
```

9. Z′yyddmm′ to C′yyyymmdd′

```
OUTFIL OUTREC=(1,2,Y2C,5,2,3,2,ZD,M11)
```

```
yy at p, mm at p+4, dd at p+2
```

10. P′yyddmm′ (X′0yyddmmC′) to C′yyyymmdd′

```
OUTFIL OUTREC=(1,2,Y2P,3,2,PD0,M11,2,2,PD0,M11)
```

```
yy at p, mm at p+2, dd at p+1
```

11. C′yyddd′ to C′yyyyddd′

```
OUTFIL OUTREC=(1,2,Y2C,3,3)

  yy at p, ddd at p+2
```

12. Z′yyddd′ to C′yyyyddd′

```
OUTFIL OUTREC=(1,2,Y2C,3,3,ZD,M11)

  yy at p, ddd at p+2
```

13. P′yyddd′ (X′yydddC′) to C′yyyyddd′

```
OUTFIL OUTREC=(1,1,Y2D,2,2,PD,M11)

  yy at p, ddd at p+1
```

14. C′dddyy′ or Z′dddyy′ to C′yyyyddd′

```
OUTFIL OUTREC=(4,2,Y2C,1,3)

  yy at p+3, ddd at p
```

15. P′dddyy′ (X′dddyyC′) to C′yyyyddd′

```
OUTFIL OUTREC=(2,2,Y2P,1,2,PD,M11)

  yy at p+1, ddd at p
```

16. C′yysmmsdd′ to C′yyyymmdd′

```
OUTFIL OUTREC=(1,2,Y2C,4,2,7,2)

  yy at p, mm at p+3, dd at p+6
```

17. C′mmsddsyy′ to C′yyyymmdd′

```
OUTFIL OUTREC=(7,2,Y2C,1,2,4,2)

  yy at p+6, mm at p, dd at p+3
```

18. C′ddsmmsyy′ to C′yyyymmdd′

```
OUTFIL OUTREC=(7,2,Y2C,4,2,1,2)

  yy at p+6, mm at p+3, dd at p
```

19. C′yysddsmm′ to C′yyyymmdd′

```
OUTFIL OUTREC=(1,2,Y2C,7,2,4,2)

  yy at p, mm at p+6, dd at p+3
```

20. C′yyyymmdd′ to C′yyyymmdd′

```
OUTFIL OUTREC=(1,8)

  yyyy at p, mm at p+4, dd at p+6
```

21. Z'yyyymmdd' to C'yyyymmdd'

```
OUTFIL OUTREC=(1,6,7,2,ZD,M11)

   yyyyat p, mm at p+4, dd at p+6
```

22. P'yyyymmdd' (X'0yyyymmddC') to C'yyyymmdd'

```
OUTFIL OUTREC=(1,5,PD0,M11)

   yyyy at p, mm at p+2, dd at p+3
```

23. C'ddmmyyyy' to C'yyyymmdd'

```
OUTFIL OUTREC=(5,4,3,2,1,2)

   yyyy at p+4, mm at p+2, dd at p
```

24. Z'ddmmyyyy' to C'yyyymmdd'

```
OUTFIL OUTREC=(5,4,ZD,M11,3,2,1,2)

   yyyy at p+4, mm at p+2, dd at p
```

25. P'ddmmyyyy' (X'0ddmmyyyyC') to C'yyyymmdd'

```
OUTFIL OUTREC=(3,3,PD0,M11,2,2,PD0,M11,1,2,PD0,M11)

   yyyy at p+2, mm at p+1, dd at p
```

26. C'cyymmdd' to C'yyyymmdd'

```
OUTFIL OUTREC=(1,1,CHANGE=(2,
                   C¢0¢,C¢19¢,
                   C¢1¢,C¢20¢,
                   C¢2¢,C¢21¢),
              2,6)

   c at p, yy at p+1, mm at p+3, dd at p+5
```

27. Z'cyymmdd' to C'yyyymmdd'

```
OUTFIL OUTREC=(1,1,CHANGE=(2,
                   C¢0¢,C¢19¢,
                   C¢1¢,C¢20¢,
                   C¢2¢,C¢21¢),
              2,4,6,2,ZD,M11)

   c at p, yy at p+1, mm at p+3, dd at p+5
```

28. P'cyymmdd' (X'cyymmddC') to C'yyyymmdd'

```
OUTFIL OUTREC=(1,1,CHANGE=(2,
                   B¢0000....¢,C¢19¢,
                   B¢0001....¢,C¢20¢,
                   B¢0010....¢,C¢21¢),
               1,4,PD0,M11)

   c at p, yy at p, mm at p+1, dd at p+2
```

29. C′mmddyyc′ to C′yyyymmdd′

```
OUTFIL OUTREC=(7,1,CHANGE=(2,
                    C¢0¢,C¢19¢,
                    C¢1¢,C¢20¢,
                    C¢2¢,C¢21¢),
               5,2,1,4)

  c at p+6, yy at p+4, mm at p, dd at p+2
```

30. Z′mmddyyc′ to C′yyyymmdd′

```
OUTFIL OUTREC=(7,1,CHANGE=(2,
                    X¢C0¢,C¢19¢,
                    X¢C1¢,C¢20¢,
                    X¢C2¢,C¢21¢),
               5,2,1,4)

  c at p+6, yy at p+4, mm at p, dd at p+2
```

31. X′0mmddyyc′ to C′yyyymmdd′

```
OUTFIL OUTREC=(4,1,CHANGE=(2,
                    B¢....0000¢,C¢19¢,
                    B¢....0001¢,C¢20¢,
                    B¢....0010¢,C¢21¢),
               3,2,PD0,M11,
               1,3,PD0,M11)

  c at p+3, yy at p+2, mm at p, dd at p+1
```

32. C′yymm′ to C′yyyymm′

```
OUTFIL OUTREC=(1,2,Y2C,3,2)

  yy at p, mm at p+2
```

33. Z′yymm′ to C′yyyymm′

```
OUTFIL OUTREC=(1,2,Y2C,3,2,ZD,M11)

  yy at p, mm at p+2
```

34. P′yymm′ (X′0yymmC′) to C′yyyymm′

```
OUTFIL OUTREC=(1,2,Y2P,2,2,PD0,M11)

  yy at p, mm at p+1
```

35. C′mmyy′ or Z′mmyy′ to C′yyyymm′

```
OUTFIL OUTREC=(3,2,Y2C,1,2)

  yy at p+2, mm at p
```

36. P'mmyy' (X'0mmyyC') to C'yyyymm'

```
OUTFIL OUTREC=(2,2,Y2P,1,2,PD0,M11)
```

```
yy at p+1, mm at p
```

37. C'yysmm' to C'yyyymm'

```
OUTFIL OUTREC=(1,2,Y2C,4,2)
```

```
yy at p, mm at p+3
```

38. C'mmsyy' to C'yyyymm'

```
OUTFIL OUTREC=(4,2,Y2C,1,2)
```

```
yy at p+2, mm at p
```

39. C'yywwd' to C'yyyywwd'

```
OUTFIL OUTREC=(1,2,Y2C,3,3)
```

```
yy at p, ww at p+2, d at p+4
```

40. Z'yywwd' to C'yyyywwd'

```
OUTFIL OUTREC=(1,2,Y2C,3,2,5,1,ZD,M11)
```

```
yy at p, ww at p+2, d at p+4
```

41. P'yywwd' (X'yywwdC') to C'yyyywwd'

```
OUTFIL OUTREC=(1,1,Y2D,2,2,PD,M11)
```

```
yy at p, ww at p+1, d at p+2
```

Overview of Performance and Capacity Improvements

DFSORT R13 PTF UN90139 provides **significant performance improvements** and increases the maximum amount of data DFSORT can sort, with the following enhancements:

- **Dataspace sorting**, introduced in DFSORT R12 for fixed-length records, is now available for **variable-length records** (MVS/ESA only).
- DFSORT data processing methods have been improved.
- **Dynamic storage adjustment** is a new feature that allows DFSORT to **automatically use more storage** than the TMAXLIM value for a Blockset sort application if DFSORT determines that doing so should improve performance.

  New installation option **DSA=n** has been added to allow you to specify the dynamic storage adjustment limit. n is a value in megabytes between 0 and 2000.
- The **upper limit** for the amount of **main storage** that can be specified and used by DFSORT has been raised from 32 megabytes to **2000 megabytes**. This new limit can be used in the existing **SIZE** and **MAINSIZE** installation and run-time options.

  Specifying more main storage can provide the following benefits:

  - It allows DFSORT to sort very large data sets more efficiently.
  - It allows more sort applications to be done entirely in main storage, eliminating the need for intermediate work space and greatly reducing the EXCP counts for those applications.
  - It increases the maximum amount of data DFSORT can process in a single sort application.
- The **upper limit** for the number of JCL and dynamically allocated **work data sets** that can be specified and used by DFSORT's Blockset technique has been raised from 32 to **100**. This new limit can be used in the existing **DYNALOC** and **DYNALLOC** installation and run-time options and for **SORTWKxx DD** statements.

  The use of more work data sets increases the maximum amount of data DFSORT can process in a single sort application.

DFSORT R13 PTF UN89708 implements changes to the DFSORT SVC that provide these **significant security and performance enhancements**:

- Security improvements that bring DFSORT up to **B1 security** standards.
- Caching selection enhancements that improve storage control caching performance, especially for SORTIN and SORTOUT devices.

Headings

id	File	Page	References
SM	SORT2000 SCRIPT		
		4	Sort and Merge on Two-Digit Year Dates 15
TR	SORT2000 SCRIPT		
		7	Transform Two-Digit Year Dates to Four-Digit Year Dates 21
SMEX	SORT2000 SCRIPT		
		15	Examples of How to Sort All Kinds of Date Fields 4, 4, 6
TREX	SORT2000 SCRIPT		
		21	Examples of How to Transform All Kinds of Date Fields 8, 10, 11

Processing Options

Runtime values:

- Document fileid .... SORT2000 SCRIPT
- Document type .... GDOC
- Document style .... DEFAULT
- Profile EDFPRF30
- Service Level 0029
- SCRIPT/VS Release 4.0.0
- Date 96.04.29
- Time 10:26:51
- Device 3820A
- Number of Passes 2
- Index .... NO

Formatting values used:

- Annotation .... NO
- Cross reference listing .... YES
- Cross reference head prefix only .... NO
- Dialog .... LABEL
- Duplex .... YES
- DVCF conditions file .... (none)
- DVCF value 1 .... (none)
- DVCF value 2 .... (none)
- DVCF value 3 .... (none)
- DVCF value 4 .... (none)
- DVCF value 5 .... (none)
- DVCF value 6 .... (none)
- DVCF value 7 .... (none)
- DVCF value 8 .... (none)
- DVCF value 9 .... (none)
- Explode .... NO
- Figure list on new page .... YES
- Figure/table number separation .... YES
- Folio-by-chapter .... NO
- Head 0 body text .... (none)
- Head 1 body text .... (none)
- Head 1 appendix text .... Appendix
- Hyphenation .... YES
- Justification .... NO
- Language .... ENGL
- Layout 1
- Leader dots .... NO
- Master index .... (none)
- Partial TOC (maximum level) 4
- Partial TOC (new page after) .... INLINE
- Print example id′s .... NO
- Print cross reference page numbers .... YES
- Process value .... (none)
- Punctuation move characters ....,
- Read cross-reference file .... (none)
- Running heading/footing rule .... NONE
- Show index entries .... NO
- Table of Contents (maximum level) 3
- Table list on new page .... YES
- Title page (draft) alignment .... RIGHT
- Write cross-reference file .... (none)